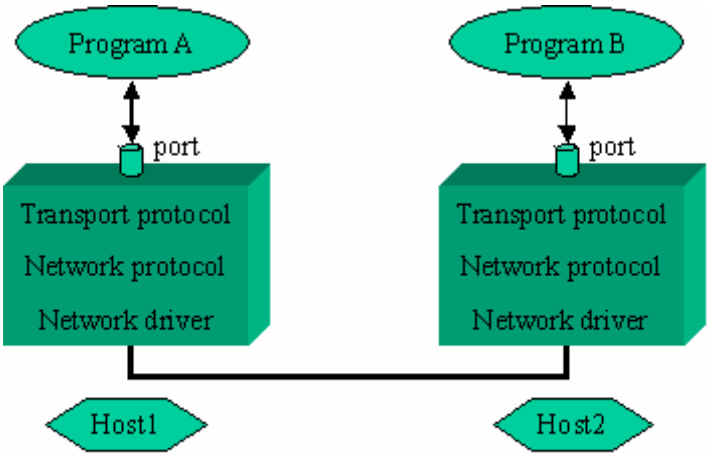
SOCKET

I. Giới thiệu

Socket là một giao diện lập trình ứng dụng (API-Application Programming Interface). Nó được giới thiệu lần đầu tiên trong ấn bản UNIX - BSD 4.2. dưới dạng các hàm hệ thống theo cú pháp ngôn ngữ C (socket(), bind(), connect(), send(), receive(), read(), write(), close() ,..). Ngày nay, Socket được hỗ trợ trong hầu hết các hệ điều hành như MS Windows (WinSock), Linux và được sử dụng trong nhiều ngôn ngữ lập trình khác nhau: như C, C++, Java, Visual Basic, Visual C++, . . .

II. Chức năng

Socket cho phép thiết lập các kênh giao tiếp mà hai đầu kênh được xác định bởi hai cổng (port). Thông qua các cổng này một tiến trình có thể nhận và gởi dữ liệu với các tiến trình khác.



III. Phân loại

Có hai kiểu socket:

1. Socket kiểu AF_UNIX chỉ cho phép giao tiếp giữa các tiến trình trong cùng một máy tính.
2. Socket kiểu AF_INET cho phép giao tiếp giữa các tiến trình trên những máy tính khác nhau trên mạng.

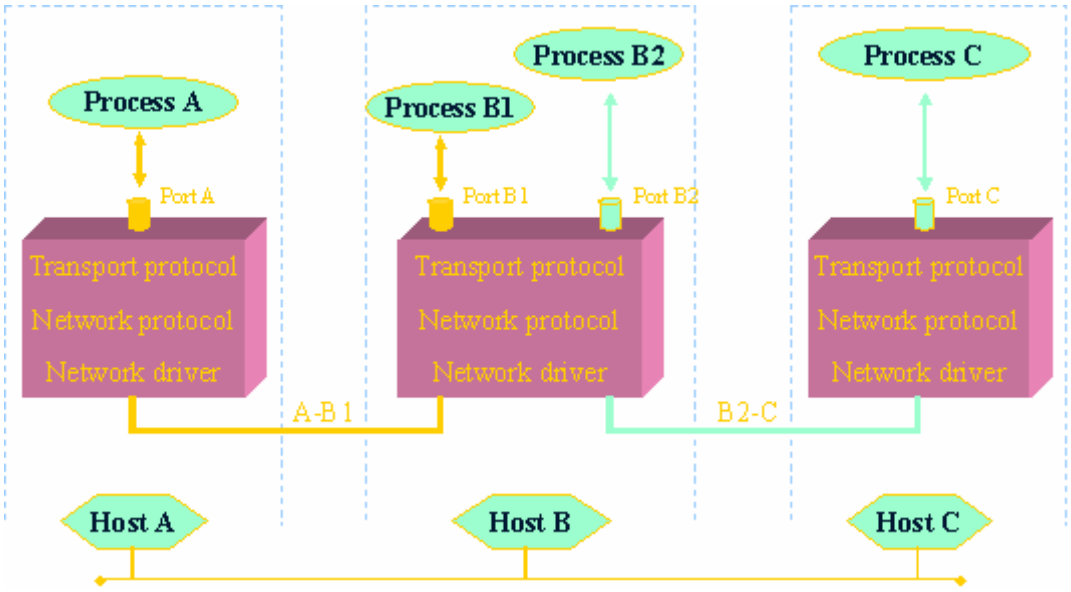
IV. Thành phần

- ***Số hiệu cổng (Port Number) của socket***

Để có thể thực hiện các cuộc giao tiếp, một trong hai tiến trình phải công bố số hiệu cổng của socket mà mình sử dụng. Mỗi cổng giao tiếp thể hiện một địa chỉ xác định trong một hệ thống. Khi tiến trình được gán một số hiệu cổng, nó có thể nhận dữ liệu gởi đến cổng này từ các tiến trình khác. Tiến trình còn lại cũng yêu cầu tạo ra một socket.

- ***Địa chỉ IP***

Ngoài số hiệu cổng, hai bên giao tiếp còn phải biết địa chỉ IP của nhau. Địa chỉ IP giúp phân biệt máy tính này với máy tính kia trên mạng TCP/IP. Trong khi số hiệu cổng dùng để phân biệt các quá trình khác nhau trên cùng một máy tính.



Trong hình trên, địa chỉ của tiến trình B1 được xác định bằng 2 thông tin: (Host B, Port B1):

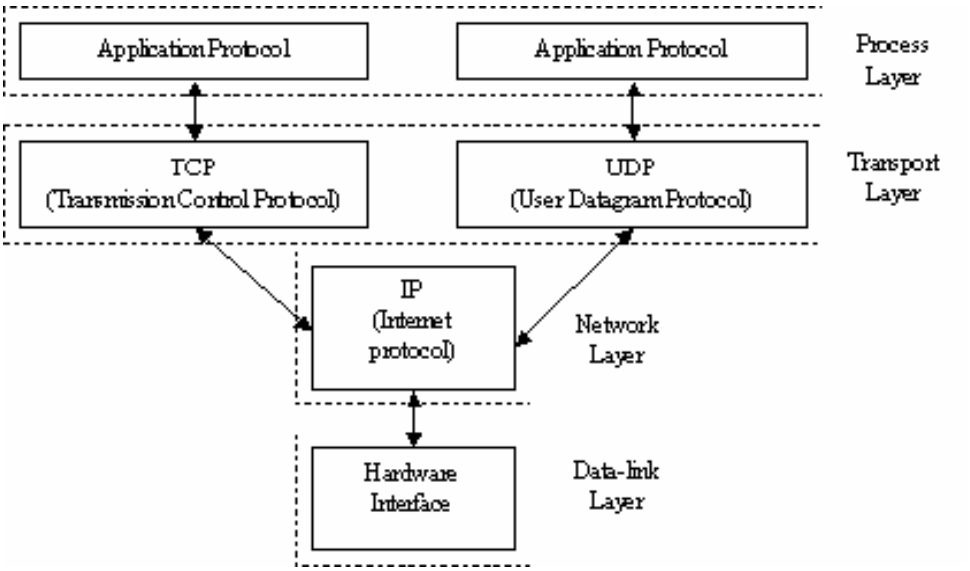
- Địa chỉ máy tính có thể là địa chỉ IP dạng 203.162.36.149 hay là địa chỉ theodạng tên miền như www.cit.ctu.edu.vn
- Số hiệu cổng gán cho Socket phải duy nhất trên phạm vi máy tính đó, có giá trị trong khoảng từ 0 đến 65535 (16 bits). Trong đó, các cổng từ 1 đến 1023 được gọi là cổng hệ thống được dành riêng cho các quá trình của hệ thống.

Số hiệu cổng	**Quá trình hệ thống**
7	Dịch Echo
21	Dịch vụ FTP
23	Dịch vụ telnet
25	Dịch vụ mail (SMTP)

81	Dịch vụ Web
110	Dịch vụ mail (POP)

V. Các chế độ giao tiếp

Xét kiến trúc của hệ thống mạng TCP/IP



Tầng vận chuyển (transport layer) giúp chuyển tiếp các thông điệp giữa các chương trình ứng dụng với nhau. Nó có thể hoạt động theo hai chế độ:

- Giao tiếp có nối kết, nếu sử dụng giao thức TCP
- Hoặc giao tiếp không nối kết, nếu sử dụng giao thức UDP

Socket là giao diện giữa chương trình ứng dụng với tầng vận chuyển. Nó cho phép ta chọn giao thức sử dụng ở tầng vận chuyển là TCP hay UDP cho chương trình ứng dụng của mình.

Bảng sau so sánh sự khác biệt giữa hai chế độ giao tiếp có nối kết và không nối kết

Chế độ có nối kết	Chế độ không nối kết
• Tồn tại kênh giao tiếp ảo giữa hai bên giao tiếp • Dữ liệu được gởi đi theo chế độ bảo đảm: có kiểm tra lỗi. truyền lại gói tin lỗi hay mất, bảo đảm thứ tự đến của các gói tin . . . • Dữ liệu chính xác, Tốc độ truyền chậm.	• Không tồn tại kênh giao tiếp ảo giữa hai bên giao tiếp • Dữ liệu được gởi đi theo chế độ không bảo đảm: Không kiểm tra lỗi, không phát hiện không truyền lại gói tin bị lỗi hay mất, không bảo đảm thứ tự đến của các gói tin . . . • Dữ liệu không chính xác, tốc độ truyền nhanh. • Thích hợp cho các ứng dụng cần tốc độ, không cần chính xác cao: truyền âm thanh, hình ảnh . . .

VI. Xây dựng ứng dụng Client-Server với Socket

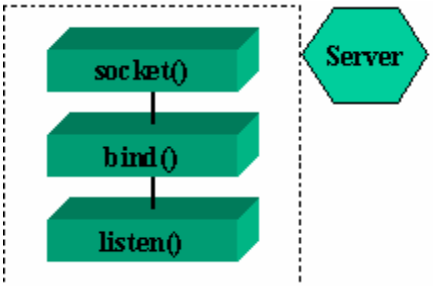
Socket là phương tiện hiệu quả để xây dựng các ứng dụng theo kiến trúc Client-Server. Các ứng dụng trên mạng Internet như Web, mail, FTP là các ví dụ điển hình.

Phần này trình bày các bước cơ bản trong việc xây dựng các ứng dụng Client-Server sử dụng Socket làm phương tiện giao tiếp theo cả hai chế độ: hướng

kết nối và hướng không kết nối.

❖ **Mô hình Client-Server sử dụng Socket ở chế độ có kết nối (TCP)**

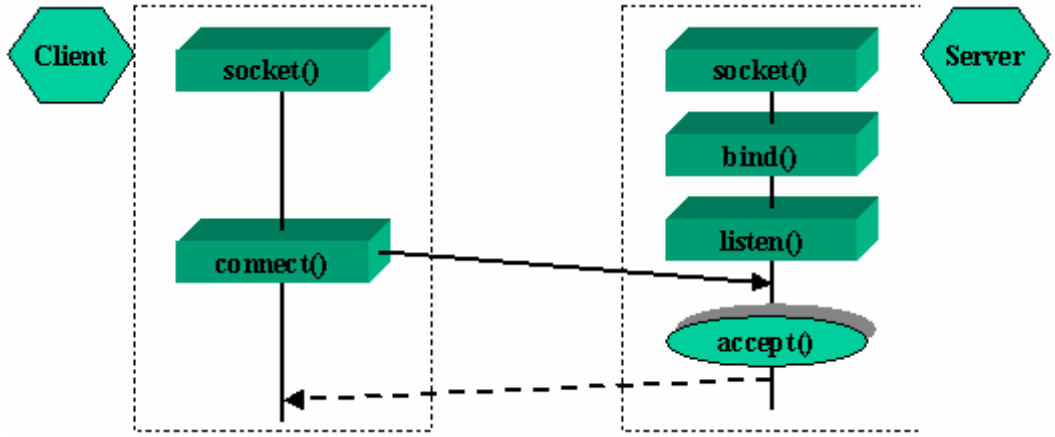
Giai đoạn 1: Server tạo Socket, gán số hiệu cổng và lắng nghe yêu cầu kết nối



- socket(): Server yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển.
- bind(): Server yêu cầu gán số hiệu port cho socket.
- listen(): Server lắng nghe các yêu cầu kết nối từ các client trên cổng đã được gán.

→ Server sẵn sàng phục vụ Client.

Giai đoạn 2: Client tạo Socket, yêu cầu thiết lập một nối kết với Server

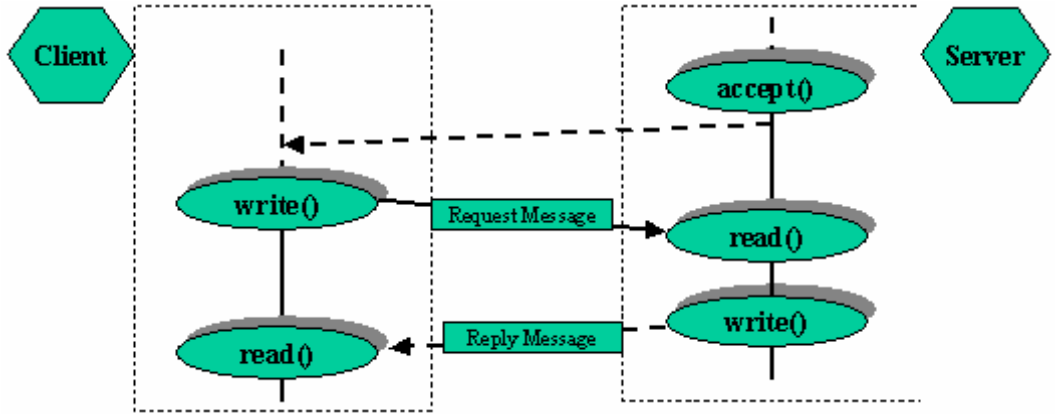


- socket(): Client yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển, thông thường hệ thống tự động gán một số hiệu cổng còn trống cho socket của Client.

- connect(): Client gởi yêu cầu nối kết đến server có địa chỉ IP và Port xác định.

- accept(): Server chấp nhận nối kết của client, khi đó một kênh giao tiếp ảo được hình thành,Client và server có thể trao đổi thông tin với nhau.

Giai đoạn 3: Trao đổi thông tin giữa Client và Server



- Sau khi chấp nhận yêu cầu nối kết, thông thường server thực hiện lệnh

read() để đợi cho đến khi có thông điệp yêu cầu (Request Message) từ client gởi đến.

- Server phân tích và thực thi yêu cầu. Kết quả sẽ được gởi về client bằng lệnh write().

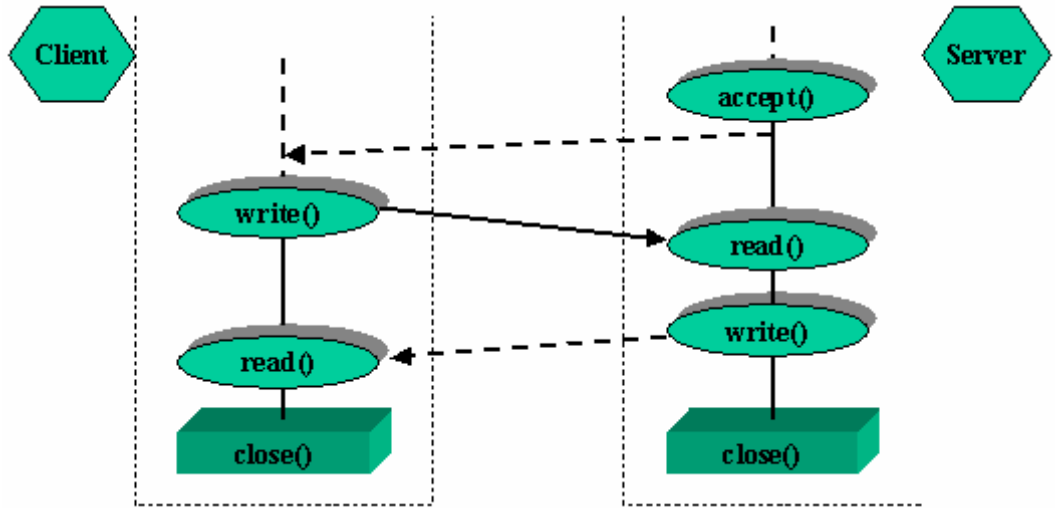
- Sau khi gởi yêu cầu bằng lệnh write(), client chờ nhận thông điệp kết quả (ReplyMessage) từ server bằng lệnh read().

Trong giai đoạn này, việc trao đổi thông tin giữa Client và Server phải tuân thủ giao thức của ứng dụng (dạng thức và ý nghĩa của các thông điệp, qui tắc bắt tay, đồng bộ hóa, ...). Thông thường Client sẽ là người gởi yêu cầu đến Server trước.

Nếu chúng ta phát triển ứng dụng theo các Protocol đã định nghĩa sẵn, chúng ta phải tham khảo và tuân thủ đúng những qui định của giao thức. Bạn có thể tìm đọc mô tả chi tiết của các Protocol đã được chuẩn hóa trong các tài liệu RFC (Request For Comments).

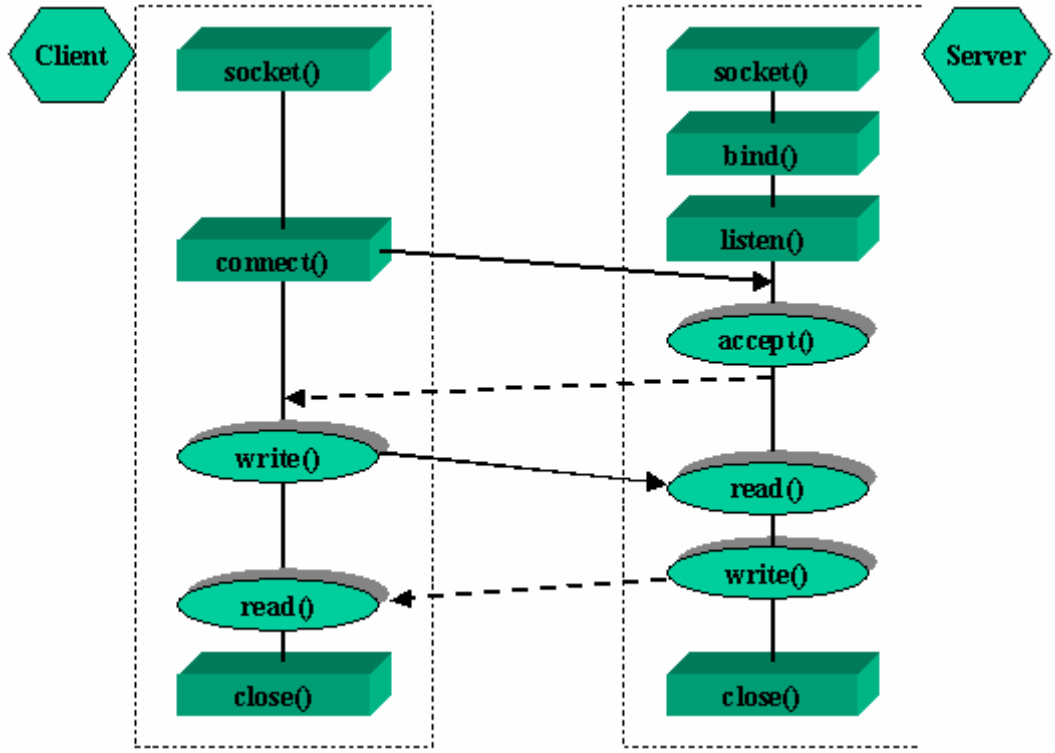
Ngược lại, nếu chúng ta phát triển một ứng dụng Client-Server riêng của mình, thì công việc đầu tiên chúng ta phải thực hiện là đi xây dựng Protocol cho ứng dụng.

Giai đoạn 4: Kết thúc phiên làm việc



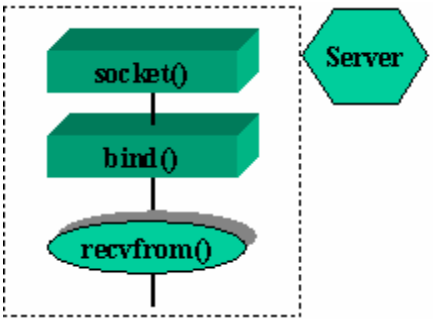
- Các câu lệnh read(), write() có thể được thực hiện nhiều lần (ký hiệu bằng hình elipse).
- Kênh ảo sẽ bị xóa khi Server hoặc Client đóng socket bằng lệnh close().

Như vậy toàn bộ tiến trình diễn ra như sau:



❖ **Mô hình Client-Server sử dụng Socket ở chế độ không nối kết (UDP)**

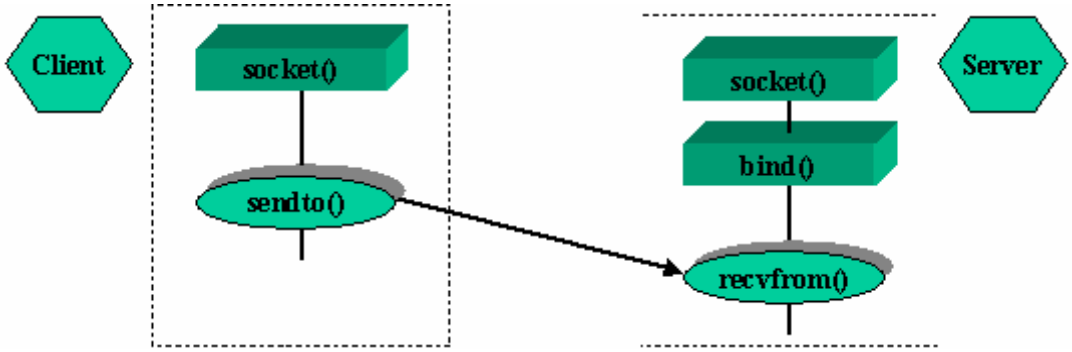
Giai đoạn 1: Server tạo Socket - gán số hiệu cổng



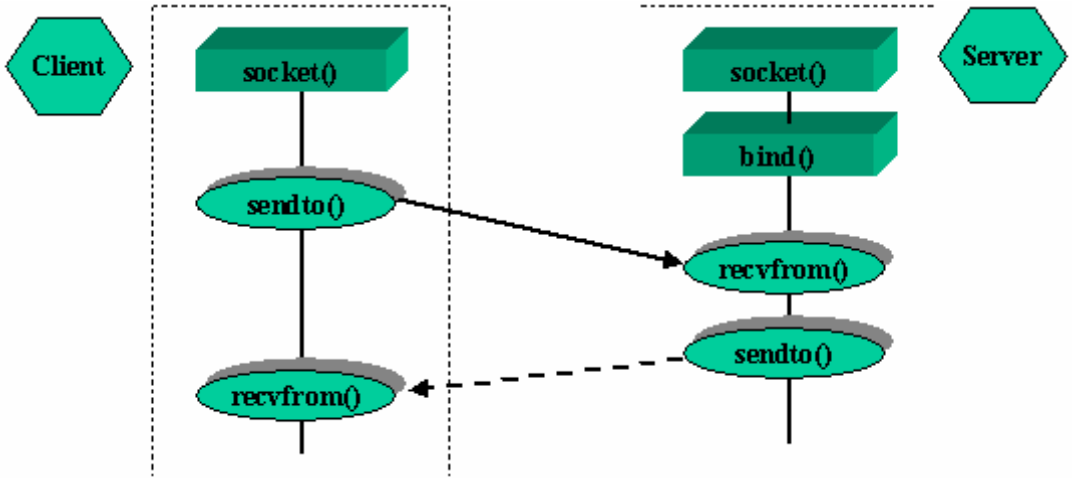
- socket(): Server yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển.

- bind(): Server yêu cầu gán số hiệu cổng cho socket..

Giai đoạn 2: Client tạo Socket



Giai đoạn 3: Trao đổi thông tin giữa Client và Server



Sau khi tạo Socket xong, Client và Server có thể trao đổi thông tin qua lại với nhau thông qua hai hàm sendto() và recvfrom(). Đơn vị dữ liệu trao đổi giữa Client và Server là các **Datagram Package** (Gói tin thư tín). Protocol của ứng dụng phải định nghĩa khuôn dạng và ý nghĩa của các Datagrame Package. Mỗi Datagrame Package có chứa thông tin về địa chỉ người gởi và người nhận (IP, Port).

— Hết —